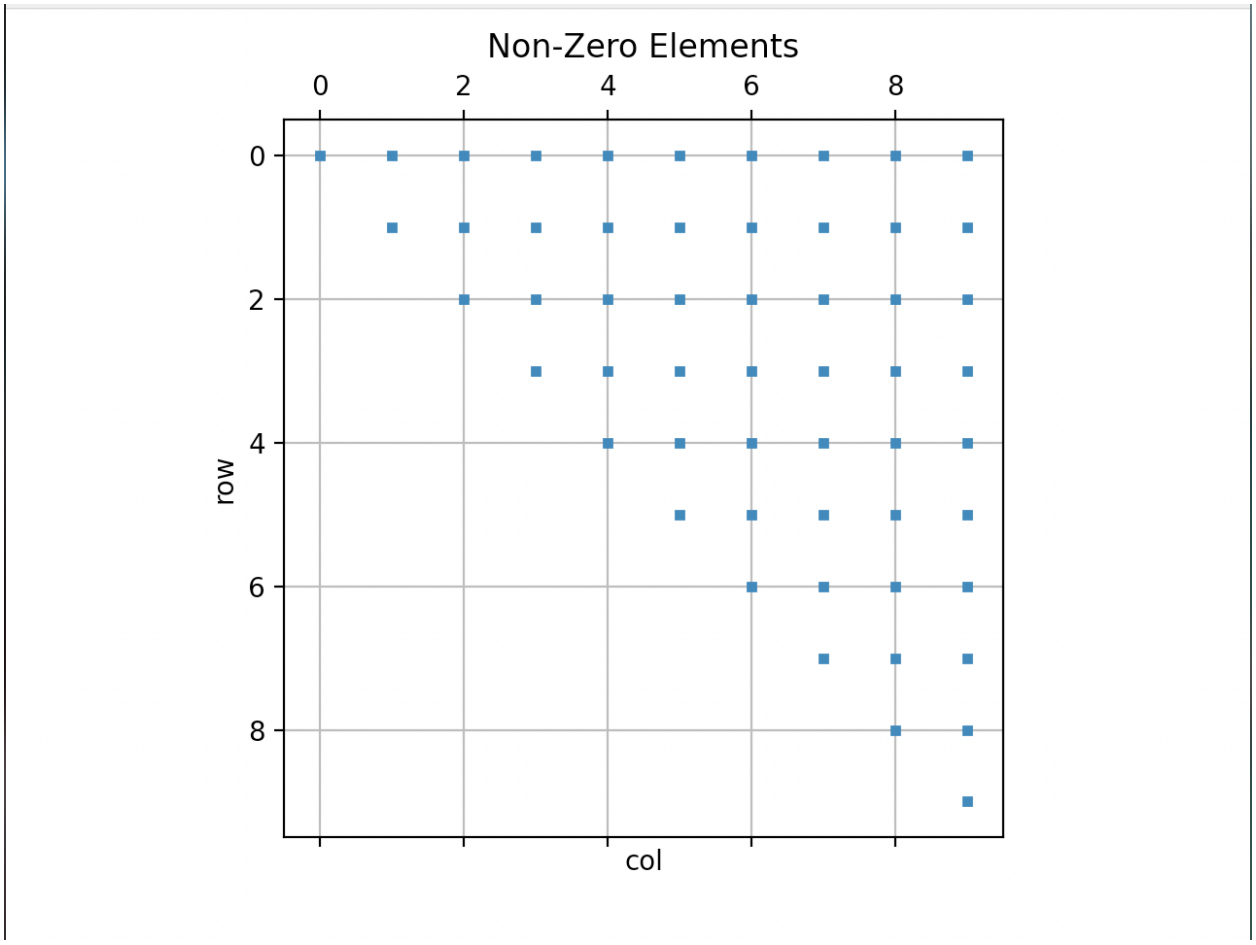
Numerical Computing  - CSCI 3656--001
Homework 4 - 09/24/21
Kai Handelman

## Matrix 1:

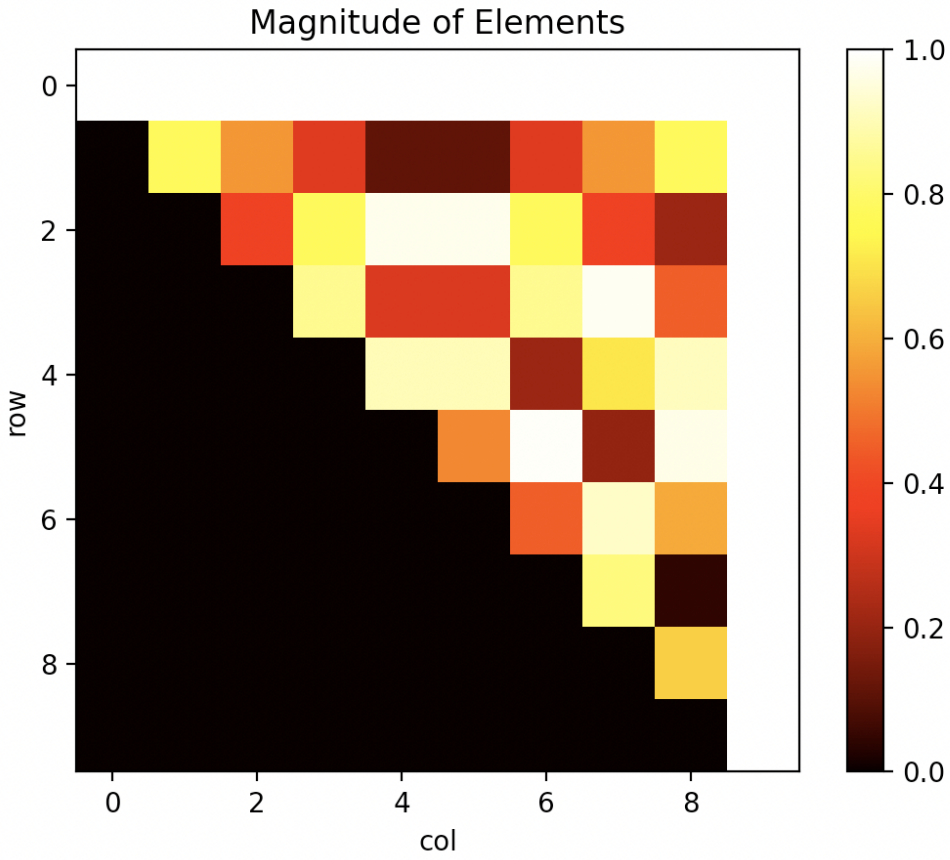
Data:

```
Data for: mat1.txt
Matrix Dimentions : 10 by 10
Number of Non Zero Elements: 55
Is Symmetrical: False
Is Diagonal: False
Is Orthogonal: False
Rank of Matrix: 10
Smallest Singular Value: 0.03001825292422508
Largest Singular Value: 3.7342634208681367
Condition Number: 124.39975871662216
No problem finding a solution 5 randomly generated right-hand-sides (i.e. b)
```

Non-Zero Visualization:




Magnitude of Elements Visualizaiton:

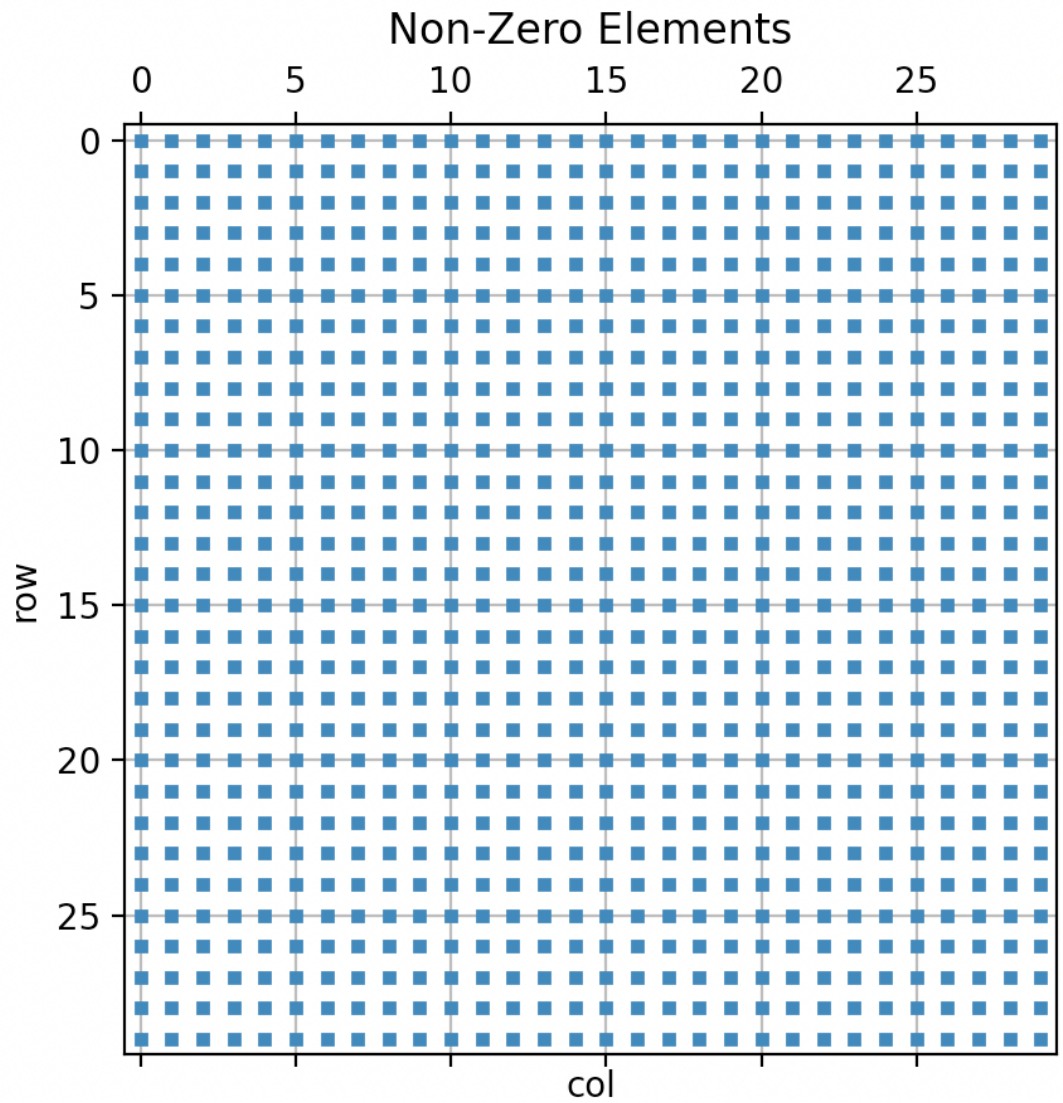

Magnitude of Elements
row
col
0
2
4
6
8
1.0
0.8
0.6
0.4
0.2
0.0

Matrix 2:

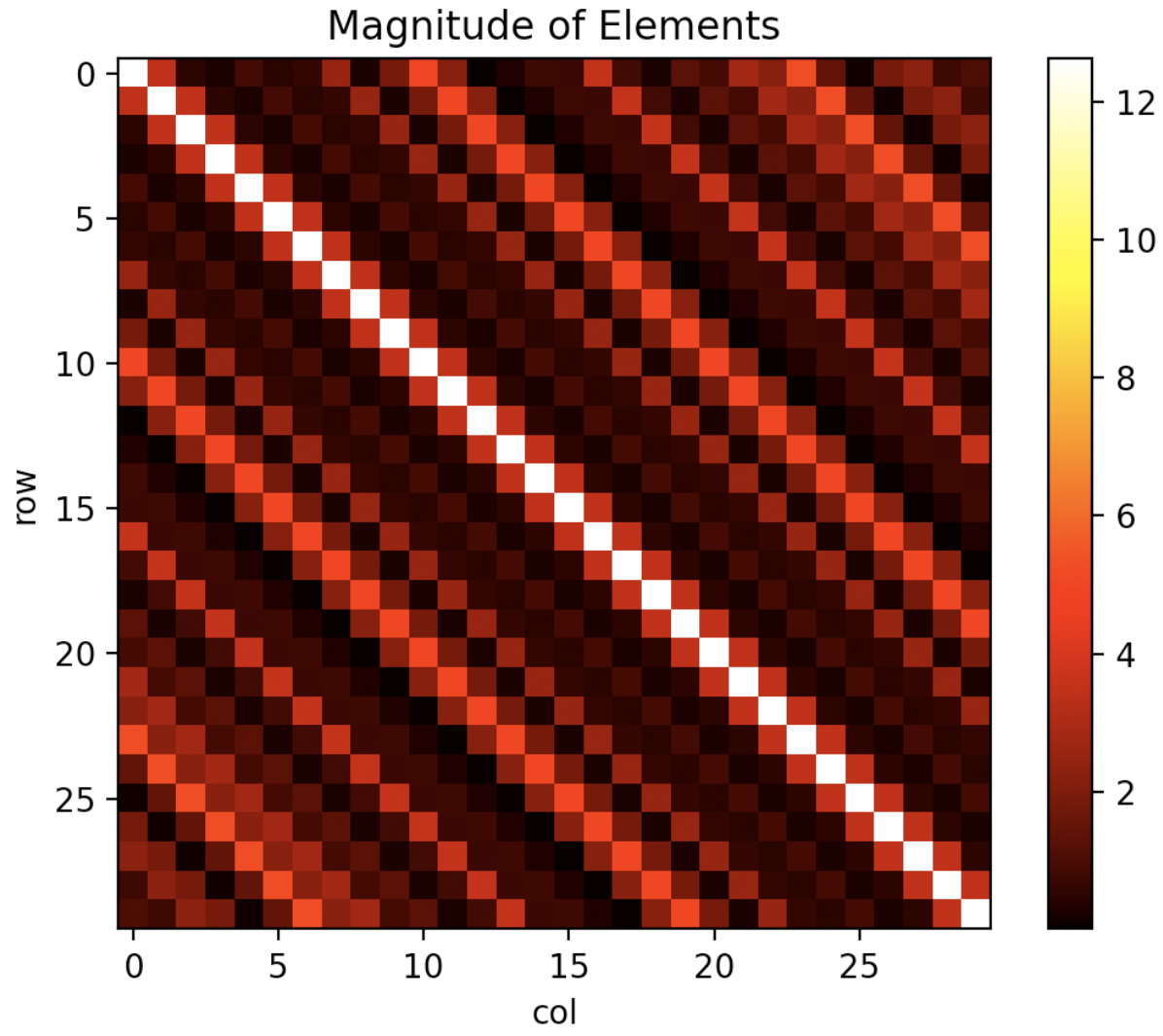
Data:

```
Data for: mat2.txt
Matrix Dimentions : 30 by 30
Number of Non Zero Elements: 900
Is Symmetrical: True
Is Diagonal: False
Is Orthogonal: False
Rank of Matrix: 30
Smallest Singular Value: 0.19847856546217035
Largest Singular Value: 41.02009374387965
Condition Number: 206.6726633597015
No problem finding a solution 5 randomly generated right-hand-sides (i.e. b)
```

Non-Zero Visualization:



Magnitude of Elements Visualizaiton:



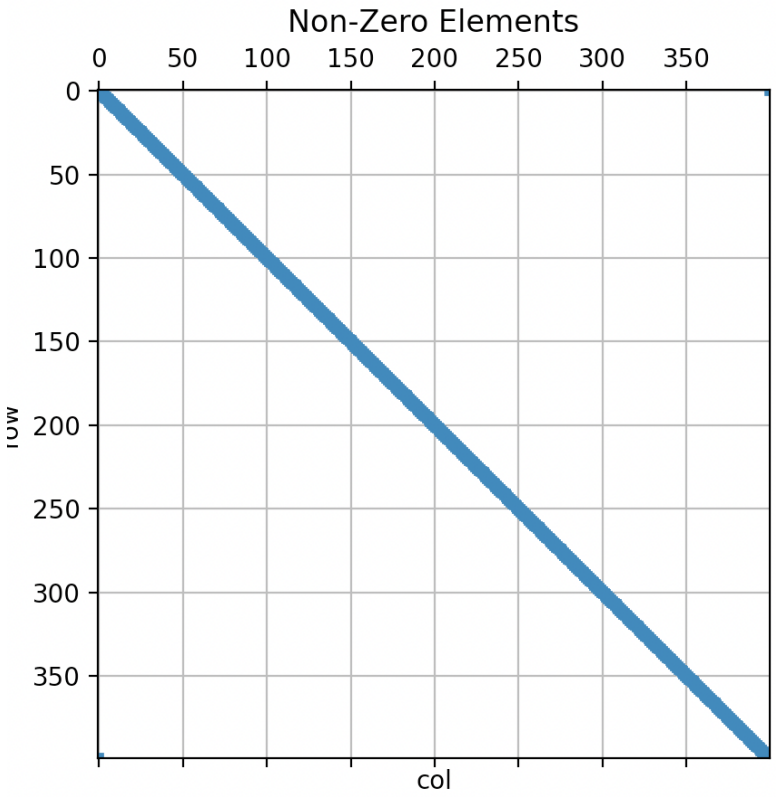
Matrix 3:

Data:

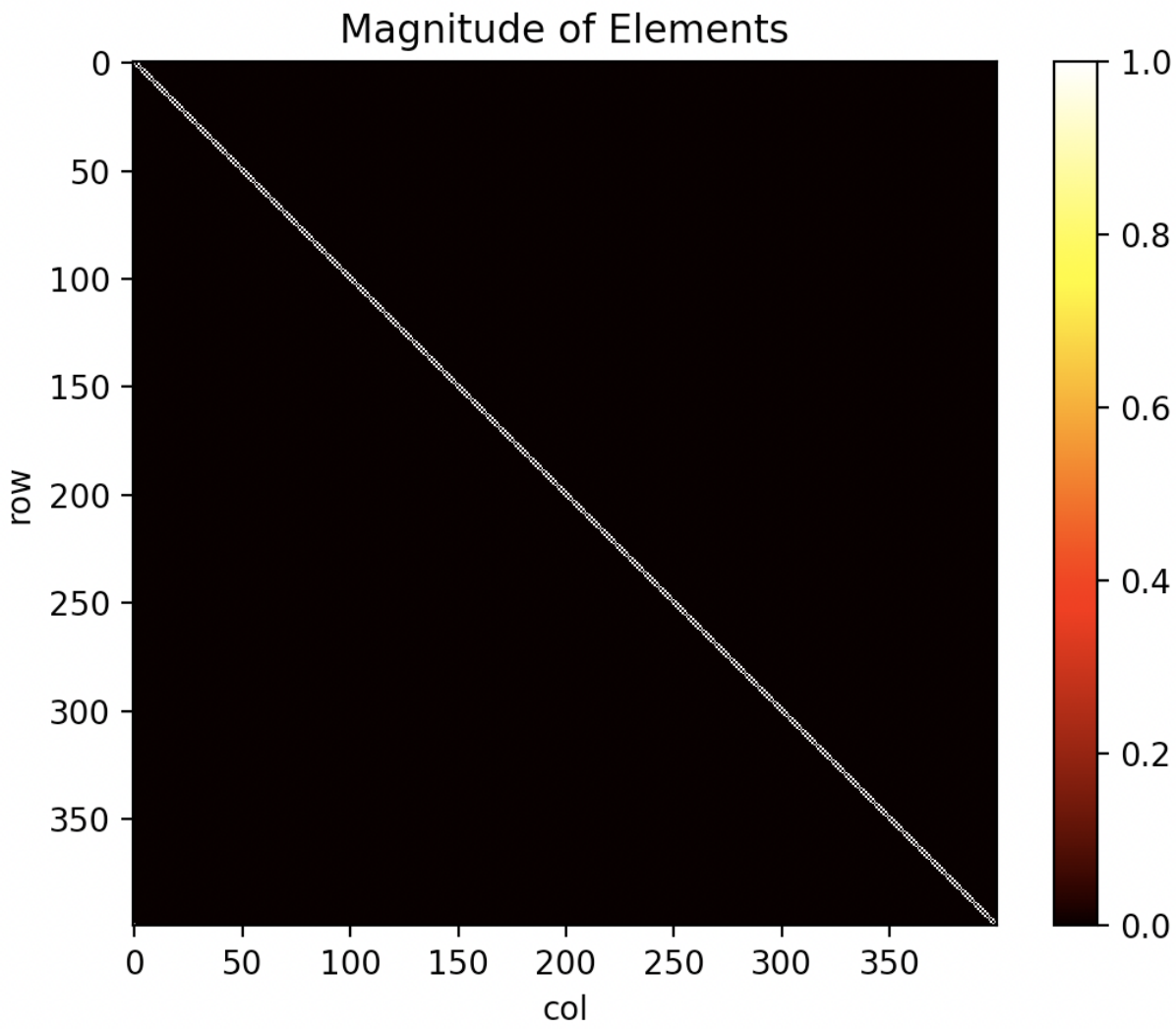
```
Data for: mat3.txt
Matrix Dimentions : 400 by 400
Number of Non Zero Elements: 800
Is Symmetrical: False
Is Diagonal: False
Is Orthogonal: False
Rank of Matrix: 399
Smallest Singular Value: 2.4510929487590 25e-16
Largest Singular Value: 1.999999999999999
Condition Number: 7.26581623654531 4e+16

Error - Couldn't find a valid x
```

Solver had issues finding a solutions

Non-Zero Visualization:




Magnitude of Elements Visualizaiton:

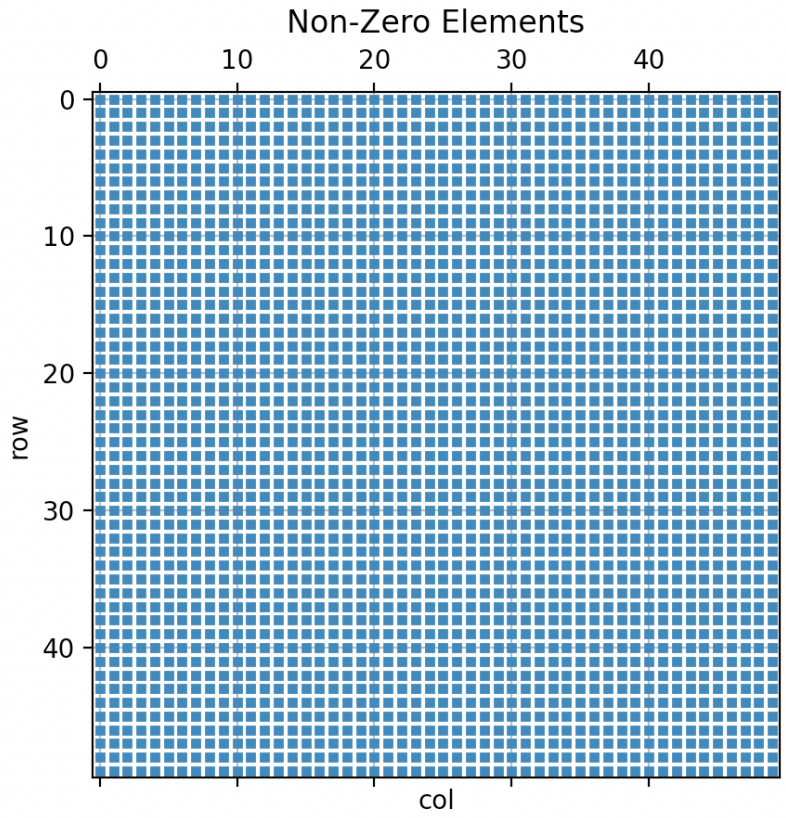

Magnitude of Elements
row
col
0
50
100
150
200
250
300
350
1.0
0.8
0.6
0.4
0.2
0.0

## Matrix 4:

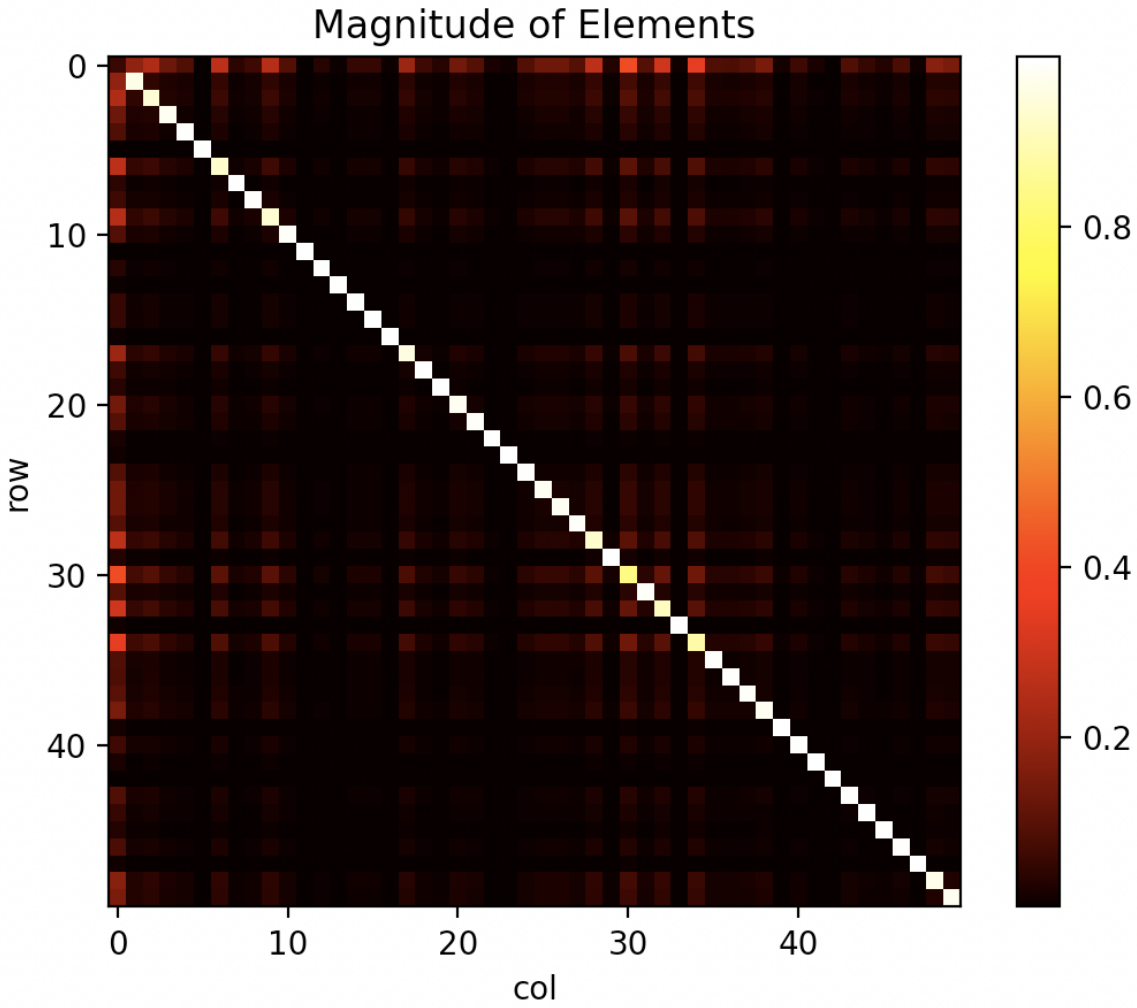
Data:

```
Data for: mat4.txt
Matrix Dimentions : 50 by 50
Number of Non Zero Elements: 2500
Is Symmetrical: False
Is Diagonal: False
Is Orthogonal: False
Rank of Matrix: 50
Smallest Singular Value: 0.9999999999999993
Largest Singular Value: 1.0000000000000009
Condition Number: 1.0000000000000016
No problem finding a solution 5 randomly generated right-hand-sides (i.e. b)
```

Non-Zero Visualization:




Magnitude of Elements Visualizaiton:

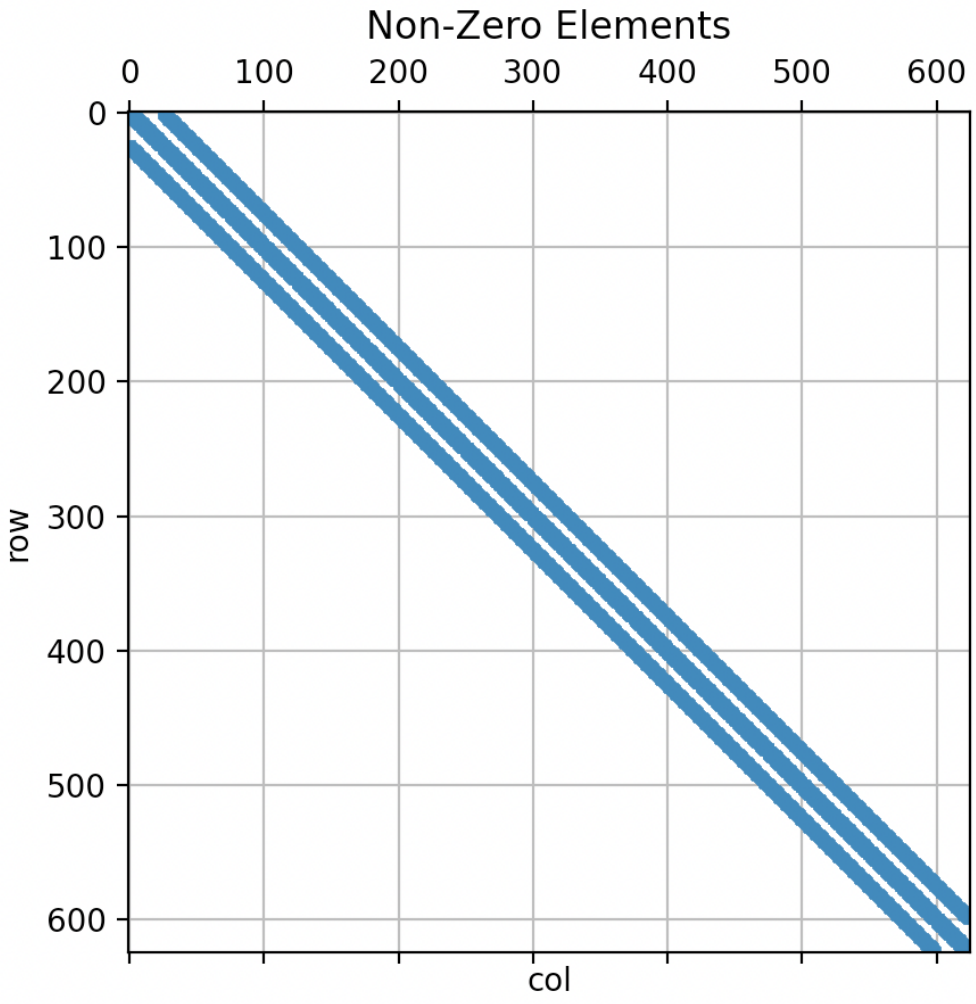

Magnitude of Elements
row
col
0
10
20
30
40
0.2
0.4
0.6
0.8

## Matrix 5:

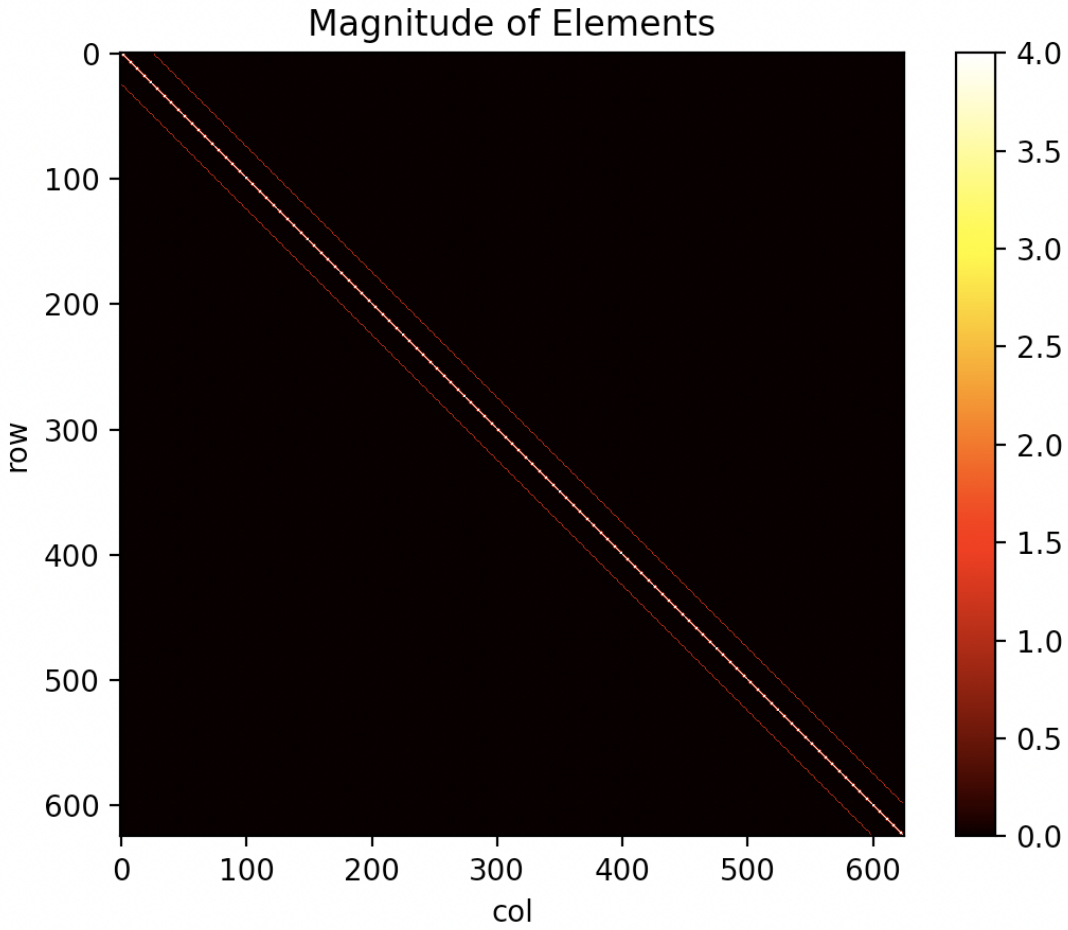
Data:

```
Data for: mat5.txt
Matrix Dimentions : 625 by 625
Number of Non Zero Elements: 3025
Is Symmetrical: True
Is Diagonal: False
Is Orthogonal: False
Rank of Matrix: 625
Smallest Singular Value: 0.029164503607784067
Largest Singular Value: 7.970835496392216
Condition Number: 273.30605737670675
No problem finding a solution 5 randomly generated right-hand-sides (i.e. b)
```

Non-Zero Visualization:




Magnitude of Elements Visualizaiton:


Magnitude of Elements
row
col
0
100
200
300
400
500
600
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0

Code:

```
def readFile(fileName):
    returnContent = []
    with open(fileName,'r') as f:
        contents = f.readlines()
        for i in contents:
            returnContent.append(i.split(","))
        for i in returnContent:
            i[len(i)-1] = i[len(i)-1].replace("\n","")

        return returnContent

def helper():...

def getTranspose(matrix):
    tMatrix = [[0 for j in range(len(matrix[0]))] for i in range(len(matrix))]
    for i in range(len(matrix)):
        for j in range(len(matrix[0])):
            tMatrix[j][i] = matrix[i][j]
    return tMatrix

def isSymmetrical(matrix):
    if len(matrix) != len(matrix[0]):
        return False
    tMatrix = getTranspose(matrix)
    for i in range(len(matrix)):
        for j in range(len(matrix[0])):
            if matrix[i][j] != tMatrix[i][j]:
                # print(i,j)
                # print(matrix[i][j])
                # print(tMatrix[i][j])
                # print(matrix[j][i])
                return False
    return True
```

```
def isDiagonal(matrix):
    for i in range(len(matrix)):
        for j in range(len(matrix[0])):
            if i != j and matrix[i][j] != 0:
                return False
    return True

def isOrthogonal(matrix):
    if len(matrix) != len(matrix[0]):
        return False
    tMatrix = getTranspose(matrix)                                  #Transposed Matrix
    rMatrix = np.matmul(matrix,tMatrix)                             #Result Matrix
    for i in range(len(matrix)):
        if rMatrix[i][i] != 1:
            return False
    return isDiagonal(rMatrix)

def getSVD(matrix):
    u,s,h = svd(matrix)
    return min(s),max(s)

def generateRight(matrix):
    n = len(matrix)
    m = 1
    b = []
    for i in range(5):
        b = np.random.rand(n,m)
        try:
            x = solve(matrix,b)
            # print("{}. b = {}".format(i+1,b))
            # print("X found: {}".format(x))
        except:
            print("\nError - Couldn't find a valid x")
            return
    print("No problem finding a solution 5 randomly generated right-hand-sides (i.e. b)")
```

```
def nonZeroGraph(matrix):
    plt.spy(matrix,markersize='3')
    plt.grid('on')
    plt.title("Non-Zero Elements")
    plt.xlabel("col")
    plt.ylabel("row")
    plt.show()

def magHeatMap(matrix):
    plt.imshow(abs(matrix),cmap="hot",interpolation='nearest')
    plt.title("Magnitude of Elements")
    plt.xlabel("col")
    plt.ylabel("row")
    plt.colorbar()
    plt.show()

    pass


def mainHwFunction(fileName):
    contents = readFile(fileName)
    print("Data for: " + fileName)
    print("Matrix Dimentions : {} by {}".format(len(contents),len(contents[0])))        #Q1: Matrix Dimensions
    matrix = np.array(contents,dtype=float)
    print("Number of Non Zero Elements: {}".format(len(matrix[matrix != 0])))           #Q2: Zero Elements
    print("Is Symmetrical: {}".format(isSymmetrical(matrix)))                           #Q3: Symmetry
    print("Is Diagonal: {}".format(isDiagonal(matrix)))                                 #Q4: Diagonal
    print("Is Orthogonal: {}".format(isOrthogonal(matrix)))                             #Q5: Orthogonal
    print("Rank of Matrix: {}".format(matrix_rank(matrix)))                             #Q6: Rank
    min,max = getSVD(matrix)
    print("Smallest Singular Value: {} \nLargest Singular Value: {}".format(min,max))   #Q7&8: SVD
    print("Condition Number: {}".format(cond(matrix)))                                  #Q9: Condition Number
    generateRight(matrix)                                                               #Q10: Solving for a random b  (Ax = b)
    nonZeroGraph(matrix)
    magHeatMap(matrix)



mainHwFunction("mat5.txt")
```